Table of Contents

Columbia

Columbia: Introduction

Columbia, an SGI Altix supercomuter named to honor the crew of Space Shuttle Columbia flight STS-107, has been in production since 2004. In March 2008, the system had 14,136 cores in 24 nodes (Columbia1-Columbia24). When the Pleiades system came into production, the original 20 Columbia nodes (1-20) were retired. Columbia currently comprises 1 front-end node (cfe2) and 4 compute nodes (Columbia21-Columbia24).

The following few articles provide Columbia hardware information at varying levels of detail:

Columbia Hardware Overview provides a high-level overview of the Columbia system architecture, including resource summaries of the compute- and front-end nodes, the interconnect, and storage capacity.

Columbia Configuration Details focuses on more detailed configuration statistics of the processors and their associated memory.

The article Columbia Home Filesystem - provides information on the quota and backup policies on the home filesystem.

The article Columbia CXFS Filesystems - details the configurations of the CXFS filesystems and users' quotas on these filesystems.

In addition, the article Columbia Front-End Usage Guidelines provides guidelines on using the front-end node (cfe2).

Columbia Hardware Overview

DRAFT

This article is being reviewed for completeness and technical accuracy.

Columbia Supercomputer

The Columbia supercluster, which ranked 2nd (51.87 Tflops/s) in the Nov 2004 Top500 list, has been in service for many years. Most of the earlier Columbia nodes (Columbia1 - Columbia20) have been retired. The remaining Columbia nodes (Columbia21-24) continue to serve the NASA community to achieve breakthroughs in science and engineering for the agency's missions and vision for Space Exploration.

Current Columbia System Facts

Manufacturer - SGI

List of nodes for Columbia system

Nodes	Type	Speed	Cache
1 Altix 4700 (512 cores)	Montecito	1.6 GHz	9MB
1 Altix 4700 (2048 cores)	Montecito	1.6 GHz	9MB
2 Altix 4700 (1024 cores)	Montvale	1.6 GHz	9MB
4 Total Compute Nodes (4,608 Total Cores)			

System Architecture

- 40 compute node cabinets
- 30 teraflop/s theoretical peak (original 10,240 system: 63 Tflop/s)

Subsystems

- 1 front-end node

Memory

- Type - double data rate synchronous dynamic random access memory (DDR SDRAM)
- Per Processor (core) - 2GB
- Total Memory - 9TB

Interconnects

- SGI® NUMAlink® interconnected single-system image compute nodes
- Internode
  - ♦ InfiniBand® - 4x (Single Data Rate, Double Data Rate)
  - ♦ 10Gb Ethernet LAN/WAN interconnect
  - ♦ 1Gb Ethernet LAN/WAN interconnect

Storage

- Online - DataDirect Networks® & LSI® RAID, 1PB (raw)
  - ♦ 1 SGI CXFS domains
  - ♦ Local SGI XFS fileystems
- Archival - Attached to high-end computing SGI CXFS SAN filesystem

Operating Environment

- Operating system - SUSE Linux Enterprise
- Job Scheduler - PBS®
- Compilers - C, Intel Fortran, SGI MPT

Columbia Configuration Details

DRAFT

This article is being reviewed for completeness and technical accuracy.

Current Columbia compute nodes, Columbia21-24, are SGI Altix 4700 systems. Detailed information about the processor and memory subsystems of these compute nodes are provided in this article.

Processor and Memory Subsystems Statistics

Below are configuration statistics for the processor and memory subsystems for Columbia21-24:

Columbia Processor and Memory Subsystems Statistics

Hostname	**Columbia21**	**Columbia22**	**Columbia23-24**
Function	compute	compute	compute
Architecture	Altix 4700 (bandwidth configuration)	Altix 4700 (density configuration)	Altix 4700 (density configuration)
	Dual-Core Processor		
Processor	**Itanium2 9040 (Montecito)**	**Itanium2 9040 (Montecito)**	**Itanium2 9150M (Montvale)**
Core-Clock	**1.6 GHz**	**1.6 GHz**	**1.67 GHz**
# of Cores/Node	2	4	4
Nodes/Blade	1	1	1
Total # of Blades	**256**	**512**	**256**
Total # of Cores	**512**	**2048**	**1024**
	Memory		
Local Memory/Node (2 Cores for C21 and 4 Cores for C22, C23-24)	**~3.8 GB**	**~7.6 GB**	**~7.6 GB**
Total Memory	~ 1000 GB	~ 4000 GB	~ 2000 GB
L1 Cache Size/Core	32KB (split into instruction and data cache)	32KB (split into instruction and data cache)	32KB (split into instruction and data cache)
L1 Cache Associativity	4-way	4-way	4-way
L1 Cache Line Size	64 bytes	64 bytes	64 bytes
L2 Cache Size/Core	1MB: instructions 256KB: data	1MB: instructions 256KB: data	1MB: instructions 256KB: data

L2 Cache Associativity	8-way	8-way	8-way
L2 Cache Line Size	128 bytes	128 bytes	128 bytes
L3 Cache Size/Core	9MB	9MB	9MB
L3 Cache Associativity	9-way	9-way	9-way
Default Page Size	16 KB	64 KB	16 KB

Itanium-64 Processors Facts

- The Itanium chip is based on the IA-64 (Intel Architecture, 64 bit) architecture that implements the EPIC (Explicit Parallel Instruction set Computing) technology. With EPIC, an Itanium processor family compiler turns sequential code into parallelized 128-bit bundles that can be directly or explicitly processed by the CPU without having to interpret it further. This explicit expression of parallelism allows the processor to concentrate on executing parallel code as fast as possible, without further optimizations or interpretations. On the contrary, a regular (non-Itanium's processor family) compiler takes a sequential code and examines and optimizes it for parallelism, but then has to regenerate sequential code in a such a way that the processor can re-extract the parallelization from it. The processor then has to read this implied parallelism from the machine code, re-build it, and run it. The parallelism is there, but it is not as obvious to the processor, and more work has to be done by the hardware before it can be utilized.
- Unlike the RISC processors (as used in the SGI Origins) that dedicate an enormous amount of chip real estate and logic to hide cache misses (by allowing instructions to be executed out of order, which works well when the ratio of CPU frequency to memory frequency is relatively small), the EPIC processors rely on the software to make sure that the data is in the proper cache at the proper time. Instructions are issued in order, so there is no hardware mechanism to hide a cache miss.
- The Itanium processors use long instruction words. Specifically, three instructions are grouped into a 128-bit bundle. Each instruction is 41 bits wide. The least significant 5 bits encode a bundle template. The template field encodes (1) the execution units (integer units I, memory units M, floating point units F, and branch units B) needed by the three instructions, and (2) which instructions can be executed in parallel. For the Itanium2 chips, two bundles can be executed per cycle.
- Four memory-load operations per cycle can be delivered from the L2 cache to the floating-point register file. This will completely support two floating-point operations per cycle; this translates into **4 FLOPS per cycle** using the FMA operation.
- Branch predication: without predication, parallelism would be impossible. Instead of waiting for each section of a complex calculation to finish, it is faster if the processor can predict the outcome and proceed on the basis of that prediction. These prediction points are called branches, and current processors try to guess which branch to take. If it predicts correctly, the whole calculation is validated. If it predicts incorrectly, the string has to be thrown out and the calculation starts over. The Itanium processor family architecture minimizes wasted calculations by taking both possible paths to the next branch, where it follows both branches again. When it comes to the correct result it drops the other branch path that it doesn't need, keeps

the branch that it does and it continues on with the calculation.

- Speculative loads; a processor needs to access the memory to get code to execute, but while it fetches this code it is not executing instructions. A processor based on the Itanium processor family architecture specification can look ahead at its instruction and load the required data from the memory early; so, when those instructions begin to execute, they have the required data, even if the loaded data changes.
- 128 integer registers; up to 96 rotating

  Note: 32 registers are fixed and 96 are "stacked". A procedure call can allocate up to 96 of the stacked registers and still has access to the 32 common registers. Each procedure has its own register frame, which is flexible in size. Since most procedure calls will allocate only a few new registers, many calls can be made before the physical limits of the register file are exceeded. A dedicated piece of hardware called the Register Stack Engine (RSE) will quickly and automatically spill older registers to free up space in the register stack for the new request. The RSE will also restore spilled registers as needed.
- 128 floating-point registers; up to 96 rotating
- 64 1-bit predicate registers; up to 48 rotating
- 8 branch registers
- 128 application registers (for example, loop or epilog counters for loop optimization)
- Performance Monitor Unit (PMU)
- Advanced Load Address Table (ALAT) ALAT keeps track of speculative, or advance loads. However, an excessive number of ALAT comparisons that result in a failed advance load will seriously degrade performance.
- 3 predicated instructions in a single 128-bit bundle
- 2 bundles (that is, 6 instructions) per clock cycle
- 6 integer units
- 2 loads and 2 stores per clock cycle
- 11 issue ports

Main Memory - Global Shared Memory

SGI Altix systems dramatically reduce the time and resources required to run applications by managing extremely large data sets in a single, system-wide, shared-memory space called global shared memory. Global shared memory means that a single memory address space is visible to all system resources, including microprocessors and I/O, across all nodes. Systems with global shared memory allow access to all data in the system's memory directly and efficiently, without having to move data through I/O or network bottlenecks. On the contrary, clusters with multiple nodes without global shared memory must pass copies of data, often in the form of messages, which can greatly complicate programming and slow down performance by increasing the time processors must wait for data.

If an Altix system is configured as a multi-partition cluster, global shared memory can be achieved by using a sophisticated system memory interconnect like SGI's NUMAlink and application libraries that enable shared-memory calls, such as MPT and XPMEM (a driver which allows shared memory across partitions) from SGI.

To configure an Altix system as a single system image machine, special versions of a scalable operation system from SGI is used and no XPMEM is needed. The current version of the OS used is "2.6.16.60-0.42.9.1-nasa64k #1 SMP".

The SGI Altix systems use the non-uniform memory access (NUMA) model. Memory subsystems from different nodes are connected through SHUB and NUMAlink interconnects.

Latency:

The local memory latency (within a node) is about 145 nanoseconds (ns). Latency from the other node of the same C-brick is 290 ns. Each additional router hop adds 45 - 50 ns (for NUMAlink 3 protocol). Each meter of NUMAlink cable adds 10 ns.

Maximum number of router hops:

- 16 CPUs - 3 hops
- 32 CPUs - 4 hops
- 64 CPUs - 5 hops
- 128 CPUs - 5 hops
- 256 CPUs - 7 hops

Bandwidth:

The Altix memory subsystem uses PC-style double data rate (DDR) SDRAM DIMMs. Each SHUB supports four DDR buses. Each DDR bus may contain up to four DIMMs. The four memory buses are independent and can operate simultaneously to provide up to 12.8 GB/sec of memory bandwidth. (Local memory bandwidth for DIMM type PC2700 is 10.2 GB/sec; and for type PC3200, it is 12.8 GB/sec.) While the local processor bus has a peak bandwidth (between L3 cache and memory) of 6.4 GB per second, the local memory subsystem has enough bandwidth to fully saturate the local processor demands while leaving available bandwidth to service remote processor and I/O memory requests.

Columbia Home Filesystems

DRAFT

This article is being reviewed for completeness and technical accuracy.

Columbia's home fileystem (/u/username) is NFS-mounted on the Columbia front-end (cfe2) and compute nodes (Columbia21-24).

Once a user is granted an account on Columbia, the home directory is set up automatically during his/her first login.

Quota and Policy

Disk space quota limits are enforced on the home filesystem. By default, the soft limit is 4GB and the hard limit is 5GB. There are no inode limits on the home filesystem.

To check your quota and usage on your home filesystem, do:

```
%quota -v
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
  ch-rg1:/home6    4888  4000000 5000000            294       0       0
```

The quota policy for NAS states that if you exceed the soft quota, an email will be sent to inform you of your current usage and how much of your grace period remains. It is expected that a user will occasionally exceed their soft limit as needed; however after 14 days, users who are still over their soft limit will have their batch queue access to Pleiades disabled. If you believe that you have a long-term need for higher quota limits, you should send an email justification to support@nas.nasa.gov. This will be reviewed by the HECC Deputy Project Manager, Bill Thigpen, for approval.

The quota policy for NAS can be found here.

Backup Policy

Files on the home filesystem are backed up daily.

Columbia CXFS Filesystems

Columbia CXFS filesystems (/nobackup[1-2][a-i]) are shared and accessible from cfe2 and Columbia21-24. This allows user jobs to be load-balanced across Columbia's systems without forcing users to move their data to a particular Columbia system.

Users will have a nobackup directory on one of these shared file systems. To find out where your nobackup directory is, log in to the front-end node and type the following shell command:

```
cfe2% ls -d /nobackup[1-2][a-i]/$USER
/nobackup1f/username/
```

In this example, the user is assigned to /nobackup1f.

Default Quota and Policy on /nobackup

Disk space and inodes quotas are enforced on the CXFS /nobackup[1-2][a-i] filesystems. The default soft and hard limits for inodes are 25,000 and 50,000, respectively. Those for disk space are 200GB and 400GB, respectively. To check your disk space and inodes usage and quotas on your CXFS filesystem, do the following:

```
cfe2% quota -v
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
/dev/cxvm/nobackup1f
                1673856  210000000 420000000              10973   25000   50000
```

The NAS quota policy states that if you exceed the soft quota, an email will be sent to inform you of your current usage and how much of your grace period remains. It is expected that users will occasionally exceed their soft limit, as needed; however after 14 days, users who are still over their soft limit will have their batch queue access to Columbia disabled.

If you anticipate having a long-term need for higher quota limits, please send a justification via email to support@nas.nasa.gov. This will be reviewed by the HECC Deputy Project Manager for approval.

For more information, see also, Quota Policy on Disk Space and Files.

Important: Backup Policy

As the names suggest, these filesystems are not backed up, so any files that are removed *cannot* be restored. Essential data should be stored on Lou1-3 or onto other more permanent storage.

Accessing CXFS from Lou

The Columbia CXFS filesystems are also mounted on Lou1-3. This allows you to copy files between the CXFS filesystems and your Lou home filesystem, using the *cp* or *cxfscp* commands on Lou.

Columbia Front-End Usage Guidelines

DRAFT

This article is being reviewed for completeness and technical accuracy.

The front-end system, cfe2, provide an environment that allows users to get quick turnaround while performing the following: file editing; file management; short debugging and testing sessions; and batch job submission to the compute systems.

Running long and/or large (in terms of memory and/or number of processors) debugging or production jobs interactively or in the background of cfe2 is considered to be inconsiderate behavior to the rest of the user community. If you need help submitting such jobs to the batch systems, please contact a NAS scientific consultant at (650) 604-4444 or 1-800-331-USER or send e-mail to: support@nas.nasa.gov

Jobs that cause significant impact on the system load of the Columbia front-end machine (cfe2) are candidates for removal in order to bring the front-end systems back to a normal and smooth environment for all users. A *cron* job regularly monitors the system load and determines if job removal is necessary. The criteria for job removal are described below. Owners of any removed jobs will receive a notification e-mail.

1. To be eligible for removal, the number of processors a front-end interactive job uses can be one (1) or more. Exceptions to this are those programs, utilities, etc. common to users and/or NASA missions that are listed in an "exception file". Examples of these would be:

   bash cp csh emacs gzip rsync scp sftp sh ssh tar tcsh

   Users can submit program names to be added to this exception file by mailing requests to: support@nas.nasa.gov
2. For qualifying processes, the CPU time usage of each process in a job has, on the average, exceeded a threshold defined as:

   (20 min x 8 / number of processes for the job)

   That is, a baseline for removal is a job with 8 processors running for more than 20 minutes. The maximum amount of time allowed for each processor in a job is scaled using the formula:

   20 min x 8 cpu / number-of-processes

   Therefore, the following variations are possible:

   - 160 minutes = (20 * 8) / 1 cpu

- 80 minutes = (20 * 8) / 2 cpu
- 40 minutes = (20 * 8) / 4 cpu
- 20 minutes = (20 * 8) / 8 cpu
- 10 minutes = (20 * 8) / 16 cpu
- 5 minutes = (20 * 8) / 32 cpu
- 2.5 minutes = (20 * 8) / 64 cpu

The conditions of removal are subject to change, when necessary.